B1 16, 1998, now U.S. Patent Number 6,167,365 granted December 26, 2002; and--

Rewrite the paragraph at page 1, lines ~~7 to 9~~ as follows:

B2 --U.S. Patent Application Serial Number 09/483,367, entitled "EMULATION SUSPEND MODE WITH DIFFERING RESPONSE TO DIFFERING CLASSES OF INTERRUPTS" claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 10 to 11 as follows:

B3 --U.S. Patent Application Serial Number 09/481,852, entitled "EMULATION SUSPENSION MODE WITH STOP MODE EXTENSION" claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 12 to 13 as follows:

B4 --U.S. Patent Application Serial Number 09/483,568, entitled "EMULATION SUSPEND MODE HANDLING MULTIPLE STOPS AND STARTS" claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 14 to 15 as follows:

B5 --U.S. Patent Application Serial Number 06/09/483,697, entitled "EMULATION SUSPEND MODE WITH FRAME CONTROLLED RESOURCE ACCESS " claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 16 to 17 as follows:

B6 --U.S. Patent Application Serial Number 09/482,902, entitled "EMULATION SUSPEND MODE WITH INSTRUCTION JAMMING" claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 18 to 20 as follows:

B7 --U.S. Patent Application Serial Number 09/483,237, entitled "EMULATION SYSTEM WITH SEARCH AND IDENTIFICATION OF OPTIONAL EMULATION PERIPHERALS" claiming priority from U.S. Provisional Application No. 60/120,960 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 21 to 23 as follows:

B8 --U.S. Patent Application Serial Number 09/483,783, entitled "EMULATION SYSTEM WITH ADDRESS COMPARISON UNIT AND DATA COMPARISON UNIT OWNERSHIP ARBITRATION" claiming priority from U.S. Provisional Application No. 60/120,791 filed February 19, 1999;--

Rewrite the paragraph at page 1, lines 24 to 26 as follows:

B9 --U.S. Patent Application Serial Number 09/481,853, entitled "EMULATION SYSTEM WITH PERIPHERALS RECORDING EMULATION FRAME WHEN STOP GENERATED" claiming priority from U.S. Provisional Application No. 60/120,810 filed February 19, 1999; and--

Rewrite the paragraph at page 1, lines 27 to 29 as follows:

B10 --U.S. Patent Application Serial Number 09/483,321, entitled "EMULATION SYSTEM EMPLOYING SERIAL TEST PORT AND ALTERNATIVE DATA TRANSFER PROTOCOL" claiming priority from U.S. Provisional Application No. 60/120,667 filed February 19, 1999.--

Rewrite the paragraph at page 7, lines 14 to 19 as follows:

B11 --Debug host computer 1 consists of a computer, for example a PC, running a CPU core specific software debugger as one of its tasks. The debug host computer 1 allows the user to issue high level commands such as setting breakpoints, single stepping the programmable digital processor in target system 3 and displaying the contents of a memory range.--

Rewrite the paragraph at page 10, lines 6 to 17 as follows:

B2

--The preferred embodiment of this invention includes an extension to the JTAG interface. Two pins nET0 and nET1 serve as a two pin trigger channel function. This function supplements the serial access capability of the standard interface with continuous monitoring of device activity. The two added pins create debug and test capabilities that cannot be created with the standard interface. The nET0 signal is called Emulation and Test 0 Not. This signal helps create a trigger to channel zero. Similarly, the nET1 signal is called Emulation and Test 1 Not. This signal helps create a trigger to channel one. These channels will be further explained below.--

Rewrite the paragraph at page 11, lines 3 to 16 as follows:

B3

--Figure 4 illustrates an electrical connection view of the coupling between access adapter 2 and target system 3. Figure 4 shows the connections of the of the various signals of the JTAG header 5 illustrated in Figure 2. All these signals are connected to scan controller 41. The signals nTRST, TCK and TMS are connected to two example megamodules 31 and 33. Figure 4 illustrates the optional connection of TCKO to the target system clock SYSCLK. The scan input TDI connects to a scan input of megamodule 31. The scan output of megamodule 31 supplies the scan input of eg module 33. The scan output of megamodule 33 supplies the scan output TDO. The two extension signals nET0 and nET1 control meg modules 31 and 33 via merge unit 32. These extension signals are monitored by test equipment 43.--

Rewrite the paragraph at page 11, lines 17 to 24 as follows:

B4

--The debugging environment illustrated in Figures 1 to 4 permit the user to control application execution by any programmable digital processor of target system 3. Typical control processes

Received from < 972 917 4417 > at 12/20/02 3:44:21 PM [Eastern Standard Time]

B14 include: keyboard directives such as run, halt and step; software breakpoints using op-code replacement; internal analysis breakpoints specified by the program counter or watchpoints specified by data accesses; and externally generated debug events.--

Rewrite the paragraph at page 11, line 25 to page 12, line 2 as follows:

B15 --Actions such as decoding a software breakpoint instruction (DSTOP), the occurrence of an analysis breakpoint or watchpoint (ASTOP), or the occurrence of a debug host computer event (HSTOP) are referred to as debug events. Debug events cause execution to suspend. Debug events tied to the execution of specific instructions are called breakpoints. Debug events generated by memory references are called watchpoints. External debug events can also suspend execution. Debug events cause entry into the Debug State.--

Rewrite the paragraph at page 12, line 17 to page 13, line 2 as follows:

B16 --The operational state transits from execute state 101 to debug suspend state 102 upon a debug event. The debugging environment of the preferred embodiment of this invention allows the suspension of program execution at points defined by breakpoints, watchpoints, and debug software directives, provided the application is an allowable debug suspend window. In general, debug events are allowed at an instruction boundary, when reset is inactive and no interrupts are active. Program execution suspends at an instruction boundary and the operational state changes to debug suspend state 102. When any debug condition is not met, the operational state remains in execute state 101 and no debug event processing occurs. The debugging environment permits debug event processing in the delayed slots of delayed branch instructions. Debug events occurring outside the debug suspend window create a debug pending condition. This condition suspends program execution when the application enables debug interrupts by opening the debug suspend window.--

B16

Rewrite the paragraph at page 13, line 26 to page 14, line 5 as follows:

--Certain interrupts transition the operation state from debug suspend state 102 to interrupt during suspend (IDS) state 103. These interrupts are defined by a separate interrupt mask independent of the normal interrupt mask. Those interrupts defined as high priority interrupts (HPI) or foreground interrupts cause the operation state to enter the interrupt during suspend state 103 from the debug suspend state 102. The debug suspend state 102 enables high priority interrupts independent of the state of the global interrupt enable bit or of software run directives. This enables debugging of background tasks while the target device 3 continues to service a real time application via high priority interrupts.--

B17

Rewrite the paragraph at page 15, lines 1 to 9as follows:

--The digital frame counter is decremented upon each return from interrupt. This count permits the debug environment to track the status of the suspended foreground task. For example, a taken high priority interrupt may change the machine state and thus the current machine state would not reflect the suspended background task. However, if the digital frame counter were zero, then the debug environment is assured no interrupts have temporarily changed the machine state.--

B18